

Encore 154 numéros avant l'an 2000

FEVRIER 1987

Un pont vers le futur avec FORTH

EDITORIAL

Nos adhérents ont du génie, certes, mais ils tardent un peu à en faire profiter les autres. Et pour ceux qui sont à court d'idées, nous les stimulerons en adoptant un nouveau ton. Ainsi, dans l'article 'Génération de commandes FORTH depuis une chaîne de caractères', vous devrez développer vos propres routines à partir d'une idée commune. Cette formule, différente du principe de 'Questions/réponses' est destinée à vous faire travailler un peu par vous même.

Le but de cette démarche est d'obtenir diverses versions d'un même problème, donc de permettre une critique plus objective. C'est également l'occasion d'instaurer un véritable dialogue entre programmeurs, le contenu des articles se rappochant ainsi plus de la notion de forum.

On ne le répétera jamais assez, mais JEDI est avant tout votre revue. C'est vous qui en composer le contenu, nous nous occupons seulement de la mise en forme. Et peu importe le style si on dispose du support. Est-il nécessaire d'avoir un 'look pro'? Nous ne le pensons pas. Mais peut-être sommes-nous dans l'erreur.

Nous écrivons petit: c'est pour fournir plus de matière à réflexion dans le même volume. Nous ne pouvons dépasser les 100 grammes sous peine de devoir augmenter la cotisation pour cause de surtaxe postale.

Nous n'avons pas de publicité: ce n'est pas faute d'avoir essayé. Mais JEDI ne tire qu'à 500 exemplaires, donc notre impact est considéré comme négligeable.

Nous n'avons pas de sponsors, pas de subvention: peut être faut-il faire du coude à HERSANT, au risque d'être rachetés...

Nous n'avons pas de local permanent: c'est le local ou la revue. Nous ne pouvons rembourser les billets d'avions pour nos adhérents hors métropole (Argentine, Polynésie, Tunisie...).

Mais nous avons: de l'énergie à revendre, une volonté d'exister et de s'imposer, des idées et des adhérents de valeur, des sujets en or, et surtout le culot de diffuser ce que d'autres journaux considérent comme des sujets marginaux.

SOMMAIRE

ASSOCIATION JEDI 8, rue Poirier de Narçay 75014 PARIS
Tel: (1) 45.42.88.90 (de 10h à 18h)

FORTH

TROIS ROUTINES par Marc PETREMANN

LANCEMENT DE FORTH DEPUIS UN FICHIER DE COMMANDE DBASE II

Le langage FORTH et tous les logiciels créés et sauvegardés en version compilée à partir de FORTH peuvent être exécutés de manière indépendante de tous les autres langages ou progiciels. Mais il est possible, par souci de commodité de pouvoir lancer de manière automatique un programme FORTH depuis un progiciel.

Sous dBASE, on peut intégrer dans un fichier de commande (pour exemple MENU.CMD) la ligne de commande suivante:

```
QUIT TO 'B:F83','DBASE MENU.CMD'
```

en supposant que F83.COM se trouve sur le drive B. Si vous désirez conserver la mémoire dBASE en l'état, vous pouvez sauvegarder son contenu dans un fichier à part.

L'abandon de dBASE affiche

```
** fin du traitement dans dBASE version L.C.E. **
```

puis le prompt CP/M affiche

```
A>B:F83
```

et le langage FORTH démarre. Si on a la curiosité de taper DIR, on constatera qu'un fichier nommé '$$$.SUB' apparait dans la directory CP/M.

L'abandon de la session de travail sous FORTH se fait en tapant BYE. On revient sous CP/M, le prompt affichant:

```
A>DBASE MENU.CMD
```

et vous revenez sous dBASE avec démarrage automatique du fichier de commande 'MENU.CMD'.

LANCEMENT DE FORTH DEPUIS LE PROGRAMME WORDSTAR

Lorsque sous WORDSTAR, on se retrouve avec le menu des commandes WORDSTAR, différentes options sont possibles, dont l'ouverture d'un fichier, l'impression et l'exécution. Bien entendu, seul un fichier '.COM' peut être exécuté. Vous êtes sur le drive A, FORTH se trouve sur le drive B, choisir l'option L et répondre B <ret>; choisir l'option R et répondre F83 <ret>, ou encore, taper directement R et répondre B:F83 <ret>.

Une fois votre session de travail sous FORTH terminée, taper BYE, ce qui affiche:

```
Hit any key to return to WORDSTAR
```

et vous revenez directement au menu des commandes WORDSTAR. Cette option, très utile, permet de lancer des programmes tels que mise en série de la sortie imprimante:

```
STAT LST:=UL1:
```

ou même lancer dBASE...

On constate que FORTH vient compléter une panoplie d'outils en donnant accès à des ressources inhabituellement exploitées sur des programmes classiques.

GENERATION D'UNE COMMANDE FORTH A PARTIR D'UNE CHAINE DE CARACTERES

Et maintenant, dans la série "LES GRANDS TRAVAUX" voici une proposition d'étude que je vous soumet.

A ma connaissance, seuls deux langages, LISP et LOGO, sont capables d'exploiter directement le contenu d'une liste:

```
DONNES TEST" [ TD 90 AV 50 TD 90 AV 20 ]
EXEC TEST"
```

On ne peut réaliser ceci en BASIC:

```
A$="PRINT B$"
```

et ... exécuter le contenu de B$

Mais FORTH (et surtout F83), que c'est un langage très bien très joli et qui peut tout faire, accepte ce genre de manipulation après définition du mot $EXECUTE:

```
: $EXECUTE  ( adr len --)
   DUP #TIB !
    TIB SWAP MOVE
    BLK OFF   >IN OFF ;
```

puis on définit une constante chaîne:

```
: TEST  ( -- adr len)
" DARK WORDS" ;

TEST TYPE    affiche  'DARK WORDS'

TEST $EXECUTE
```

exécute un effacement d'écran et affiche le contenu du vocabulaire courant. Pour rigoler, essayez ceci:

```
: CREE
  " : TEST2 ; " $EXECUTE ;
: OUBLIE
  " FORGET TEST2 " $EXECUTE  ;
```

puis tapez

```
CREE  : T1  ;  : T2  ;  WORDS    affiche
T2  T1  TEST2 ....etc...
```

OUBLIE WORDS oublie le mot TEST2 et tous ceux définis après lui!!

Je vous laisse imaginer le parti que l'on peut tirer de $EXECUTE en traitant des chaînes de caractères par concaténation, fractionnement, évaluation, etc.... Pour exemple, imaginons de définir un mot ITEM qui recherchera la nième sous-chaîne séparée par des espaces dans une chaîne:

```
: TEST  ( -- adr long )
" WORDS DARK DIR"

TEST 1 ITEM TYPE  afficherait  'WORDS'
TEST 2 ITEM TYPE  afficherait  'DARK'
```

etc,..

et TEST 2 ITEM $EXECUTE exécute un effacement d'écran.

Nous attendons les propositions (pas toujours aux mêmes de bosser...) en respectant les contraintes suivantes:

```
STRING        n -- <mot>  en compilation
                -- adr long en exécution
```

Mot de création de variable chaîne avec n caractères réservés. A l'initialisation, long est égal à 0.

```
$!            adr1 long adr2 long2 --
```

Affecte le contenu de la chaîne 1 à la variable chaîne 2. Exemple:

```
80 STRING A$
" Ceci est un essai" A$ $!
```

```
$+            adr1 long adr2 long2 --
```

Concaténation de chaînes.

```
LEN           adr long -- long
```

Longueur d'une chaîne.

```
MID$ LEFT$ RIGHT$
```

Sans commentaire...

Bien entendu, l'exécution de ces mots doit être sécurisée. L'affectation d'une chaîne de longueur supérieure à la taille maximale affectée à une variable chaîne ne doit pas provoquer de débordement.

A partir de ces éléments, si vous vous en sentez le courage, définissez des mots capables de transformer le contenu d'une chaîne A$ du type:

```
" (A+B)*C"
```

en une chaîne B$ du type:

```
" A B + C *"
```

et qui pourra être exécuté simplement par B$ EXECUTE. A

noter que si A, B et C n'existent pas, on pourra compléter la fonction d'évaluation par l'exécution d'une chaîne C$ telle que:

```
" VARIABLE " C$ $!
B$ 1 ITEM C$ $+
C$ EXECUTE
```

Si vous y mettez de la récursivité, nous serons ravis.

ERRATA

La dernière définition de *INPUT (JEDI 32, page 2) est erronée. Voici la définition exacte:

```
: *INPUT  ( u —— d)
*OUT @
BEGIN  DUP *OUT ! OVER TIB SWAP EXPECT SPAN @ *TIB !
       BLK OFF  >IN OFF 32 WORD NUMBER? NOT
WHILE  2DROP DUP *OUT @ SWAP -
       DUP BACKSPACES DUP SPACES BACKSPACES
REPEAT
ROT DROP ROT DROP ;
```

Et pour finir, de la part de Mr JACCOMARD, les deux mots BLK>TXT et SAUVE, utilisables seulement sur la version F83 MSDOS:

- les deux mots permettant la conversion d'un fichier écran .BLK en un fichier texte, "traitable" par votre WORD ou WORDSTAR préféré. Le voici :

```
\ Conversion/transfert fichiers "texte".     19Fev87JaD
```

par A. JACCOMARD

```
SP0 @ 1280 - CONSTANT DTA      \ adr tampon transfert.
VARIABLE ADR-DTA               \ adr courante dans DTA.

: BLK>TXT    ( n°_blc -- )      \ convertit 1 bloc fich. courant.
   DTA  DUP  ADR-DTA !  B/BUF BLANK  \ tampon "blanc".
   BLOCK  ( ad_buff_disq )  16 0 DO  \ boucle pr 16 lignes.
   DUP  I C/L * +                   \ adr lig. ds buff. disque.
   ADR-DTA @  C/L  3DUP  CMOVE     ( ad_buf ad_lig ad_dta lgr )
       -TRAILING      ?DUP     IF
    +    BL     OVER C!        \ un espace de sécurité en fin ligne
    1+   2573   OVER   !       \ CRLF (0D0AH = 2573d )
    2+   ADR-DTA   !    DROP   \ mise à jour ADR-DTA et pile.
   ELSE  2DROP  THEN           \ ignore ligne vide
   LOOP   DROP ;

: SAUVE    (S -- )
 [ DOS ]
   FCB2 DUP !FCB  DUP DELETE DROP  DUP MAKE-FILE  ( fcb )
 CAPACITY 0 DO       \ pour les 'n' blocs du fich. courant.
    I  BLK>TXT       \ écrit 1 bloc "texté" de 1024 oct.
   DTA ADR-DTA @ OVER - 1+   ( fcb ad_déb lgr )
   BOUNDS  ?DO        ( ad_déb+lgr ad_déb )
   I SET-DMA  DUP     ( adr début transfert )
       WRITE
     128 +LOOP      \ écrit sur disque le bloc "texté",
  LOOP              \ par groupe de 128 oct.
     CLOSE ;

ONLY  FORTH ALSO
```

Utilisation :
 OPEN nom.BLK   (le fichier "courant" à convertir)
 SAUVE nom.DOC   (extension pour WORD; il est écrit sur le lecteur courant)

Quoi de plus simple?

PROLOG

Complément au programme PROLOG DE JEDI N° 20

FONDEMENTS

Dans toute l'histoire de la technologie MOS, concepteurs et technologues ont dû réaliser les compromis garantissant des performances et des rendements de fabrications acceptables. Le rendement théorique peut s'exprimer en fonction de la surface du circuit (S), de la densité de défauts des masques (NO) et du nombre de niveaux critiques de masques (n):

p=(1+S.NO)EXP-N

En premier lieu, on recherche la réduction de la surface, ce qui conduit à définir des dimensions minimales des motifs élémentaires, que le concepteur exploite au mieux pour réaliser la fonction au coût minimal en surface. Par ailleurs, la réduction générale des dimensions est un facteur de performances, d'où l'importance des règles de dessin, qui accompagnées par les paramètres électriques constituent les règles de conception.

Trois critères principaux permettent l'établissement des règles de dessin si le processus technologique est supposé être figé: les conditions électriques de fonctionnement (tension, courant) définissent notamment les longueurs de canaux, les possibilités de la gravure définissent les largeurs et espacements des lignes conductrices ainsi que les contacts qu'il est possible de graver, enfin les tolérances de superposition des masques définissent des gardes interniveaux. A ceci peuvent éventuellement se rajouter des configurations interdites résultant d'un non sens électrique ou d'une impossibilité technoligique.

En fait il y a une interaction profonde entre le processus technologique et les règles de conception, à un point tel qu'il est fréquent pour les circuits à production importante d'ajuster les paramètres du processus ou les dimensions des masques en vue d'optimiser le rendement pour une conception donnée.

EXEMPLE DE REGLE DE DESSIN

Les règles de dessin d'une technologie NMOS silicium ne contiennent pas moins d'une centaine de dimensions spécifiées. Elles sont généralement présentées niveau par niveau avec mention des niveaux devant respecter des contraintes par rapport au niveau considéré. La présentation de la figure ci-dessous correspond à des règles simplifiées concernant les niveaux les plus critiques, les dimensions purement indicatives correspondent à la dénomination HMOS 1:

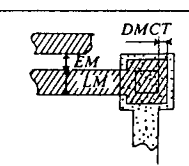
Zones diffusées		microns
largeur	LD	4
écartement	ED	5
largeur des MOS	LDM	4(1)

Précontact		
surface mini	CxC	4x4
débordement/N+	DD	1,5
débordement/Poly	DP	1,5
débordement/champ	DC	1,5

Silicium polycristallin		
largeur	LP	4
écartement	EP	4
largeur sur MOS	LPM	3,5(2)
débordement Poly/champ	DPC	3
débordement N+/Poly	DNP	3
distance N+/Poly	ENP	2

Contacts		
dimension	CTxCT	4x4
écartement	ECT	4(3)
débordement N+/CT	DNCT	2
débordement Poly/CT	DPCT	2

Aluminium		
largeur	LM	5
écartement	EM	5
débordement Al/CT	DMCT	1



Les motifs après traitement technologique diffèrent des motifs dessinés en raison de la chaîne lithographique et gravure. La différence de dimension souvent nulle peut atteindre plus ou moins un micron, avec une incertitude de l'ordre d'un demi micron. Ainsi, les motifs dont la résistance peut être affectée par des variations excessives de la géométrie seront dimensionnées à une valeur suffisante pour limiter les dispersions dans un rapport de 1 à 3 par exemple, c'est le cas de la zone diffusée (LD) ou du silicium polycristallin (LP).

Au-delà de cet aspect électrique, la limite inférieure représente celle en-dessous de laquelle les défauts de gravure introduiraient des discontinuités ou des courts circuits entre motifs. Ceci concerne d'une manière générale les autres dimensions minimales (aluminium, contacts) et les écartements. En particulier, l'écart entre zones diffusées tient compte en sus de la diffusion latérale du phénomène de perçage lié à la différence de potentiel maximale admissible entre deux diffusions.

Les tolérances de superposition des masques successifs peuvent atteindre des valeurs importantes (jusqu'à plus ou moins un micron). Toutefois, la séquence d'alignement est étudiée pour que la plupart des masques critiques soient référés au premier niveau de sorte que cette tolérance est rarement cumulée. Ceci concerne les éléments définis par la superposition de deux ou plusieurs masques: cas du précontact où la surface minimale CxC doit tenir compte des divers cas de décalage selon les différentes topologies Poly/N+, c'est le cas de l'élément MOS où le débordement de grille (DPC) est prévu pour éviter le court-circuit drain-source, tandis que le débordement N+/Poly(DNP) garantit que l'accès au transistor n'aura pas une résistance excessive.

En outre, les ouvertures de précontact et de contact ont des caratéristiques différentes. En effet, l'ouverture de précontact peut déborder sur le substrat sans inconvénient, par contre, elle doit rester éloignée de tout silicium polycristallin ou zone diffusée non équipotentielle. En ce qui concerne l'ouverture de contact Alu/Poly au Alu/N+, elle ne doit pas déborder sur le substrat en raison du risque de contact Alu/substrat (risque exclus par DNCT et DPCT), mais l'aluminium doit recouvrir entièrement l'ouverture de contact: risque de contact insuffisant ou peu fiable (DMCT).

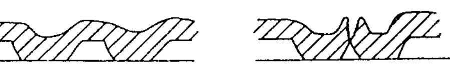
Enfin, certaines règles présentent des variantes.

1. La largeur des MOS doit être augmentée si la géométrie est critique, cela dépend notamment des tolérances de fabrication qui reproduisent les motifs à +-0,5 microns au mieux. La dimension nominale sur circuit n'est pas toujours la dimension du masque.

2. La largeur de grille définit la longueur du canal MOS (à la diffusion latérale près), ce paramètre est limité par la tenue en tension qui peut conduire à exiger des largeurs plus importantes pour les MOS à dopage superficiel plus faible (MOS déplétés). L'écartement entre zones diffusées (ED) qui obéit aux mêmes contraintes est choisi plus important car le rsique de perçage est accru sur des longues distances.

3. Généralement, la règle de dessin impose des contacts à dimension unique séparés par une distance du même ordre de préférence à un long contact rectangulaire pour des raisons de rendement de fabrication; en effet l'ouverture rectangulaire accroît les risques de décollement de la résine pendant la gravure. La séparation entre les contacts évite le profil en couteau qui risquerait de couper le conducteur d'aluminium.

Suite page 20



Séparation entre les contacts.

Décompresseur de fichiers en codage de Huffman par Linda ZANG

```
A>type UNSQZ.PAS
{$A+}
PROGRAM UNCMPRSS; { Restorer un fichier compresse }
VAR
  TABLE        : ARRAY [0..1151] OF BYTE;
  NOM0,NOM1    : STRING[16];
  CH,CH1       : BYTE;
  BUFFIN       : ARRAY [1..128] OF BYTE;
  PTFICH       : BYTE;
  FICH,FICH1   : FILE;
  PTBIT        : BYTE;      { COMPTEUR DE BITS }
  SAVEBYTE     : BYTE;      { DERNIER OCTET TROUVE }
  BYTECOUNT    : INTEGER;   { NBR. DE REPETITION DU DERNIER OCTET }
  SAVEBIT      : BYTE;      { OCTET EN COUR DE TRAITEMENT }
  I            : INTEGER;
  LENTAB       : INTEGER;   { LONGUEUR DE LA TABLE DE CODAGE }
  PTFICH1      : INTEGER;   { POINTEUR DU FICHIER OUT }
  TAMPON       : ^BYTE;     { TAMPON OUT }
  FLAG         : BYTE;      { FLAG=$FE ==> EOF }
  TAILTAMPON   : INTEGER;
  LENTAMPON    : INTEGER;
  CMDLIN       : STRING[19];
  CRC,CRC0     : INTEGER;
  UNTILFLAG    : BOOLEAN;


PROCEDURE CLOSEOUT; FORWARD;


PROCEDURE OPENIN;
BEGIN
  ASSIGN(FICH,NOM0);
  {$I-}
  RESET(FICH);
  {$I+}
  PTFICH:=129;
  PTBIT:=128;
  SAVEBYTE:=0;
  BYTECOUNT:=0;
  SAVEBIT:=0
END;


PROCEDURE READCH(VAR CH:BYTE);
BEGIN
  IF PTFICH>128 THEN
  BEGIN
    BLOCKREAD(FICH,BUFFIN,1);
    PTFICH:=1
  END;
  CH:=BUFFIN[PTFICH];
  PTFICH:=SUCC(PTFICH)
END;


FUNCTION GETBIT : BOOLEAN;
BEGIN
  IF PTBIT=128 THEN
  BEGIN
    READCH(SAVEBIT);
    PTBIT:=1
  END ELSE PTBIT:=PTBIT SHL 1;
  GETBIT:=(SAVEBIT AND PTBIT)<>0
END;


FUNCTION EVALUE : BYTE;
LABEL FIN,LOOP;
VAR   I      : INTEGER;
BEGIN
  I:=0;
LOOP:
    IF GETBIT THEN I:=I+2;
    FLAG:=TABLE[SUCC(I)];
    IF FLAG>=128 THEN GOTO FIN;
    I:=TABLE[I] SHL 2;
 GOTO LOOP;
FIN:
  EVALUE:=TABLE[I] XOR -1
END;
```

```
FUNCTION GETCHAR(VAR CH : BYTE) : BOOLEAN;
VAR
  CH1,CH2 : BYTE;
BEGIN
  IF BYTECOUNT=0 THEN
  BEGIN
    CH:=EVALUE;
    IF CH<>$90 THEN SAVEBYTE:=CH ELSE
    BEGIN
      CH1:=EVALUE;
      IF CH1=0 THEN SAVEBYTE:=CH ELSE
        IF CH1>1 THEN
        BEGIN
          BYTECOUNT:=CH1-2;
          CH:=SAVEBYTE
        END ELSE
        BEGIN
          CH:=EVALUE;
          SAVEBYTE:=CH
        END
    END
  END ELSE
  BEGIN
    CH:=SAVEBYTE;
    BYTECOUNT:=PRED(BYTECOUNT)
  END;
  IF FLAG=$FE THEN GETCHAR:=FALSE ELSE GETCHAR:=TRUE
END;


PROCEDURE OPENOUT;
BEGIN
  ASSIGN(FICH1,NOM1);
  REWRITE(FICH1);
  PTFICH1:=1;
  TAILTAMPON:=MEMAVAIL;
  IF (TAILTAMPON<0) OR (TAILTAMPON=MAXINT) THEN TAILTAMPON:=MAXINT-1;
  LENTAMPON:=TAILTAMPON DIV 128;
  TAILTAMPON:=LENTAMPON*128;
  GETMEM(TAMPON,TAILTAMPON)
END;


PROCEDURE PUTCHAR(VAR BUFF;CH:BYTE);
VAR
  BUFFOUT : ARRAY [1..1] OF BYTE ABSOLUTE BUFF;
BEGIN
  IF PTFICH1>TAILTAMPON THEN
  BEGIN
    BLOCKWRITE(FICH1,BUFFOUT,LENTAMPON);
    PTFICH1:=1
  END;
  BUFFOUT[PTFICH1]:=CH;
  PTFICH1:=SUCC(PTFICH1)
END;


PROCEDURE CLOSEOUT;
BEGIN
  PTFICH1:=PTFICH1-1;
  LENTAMPON:=PTFICH1 DIV 128;
  IF (PTFICH1 MOD 128)<>0 THEN LENTAMPON:=LENTAMPON+1;
  BLOCKWRITE(FICH1,TAMPON^,LENTAMPON);
  RELEASE(TAMPON);
  CLOSE(FICH1)
END;
```

```
BEGIN
  WRITELN;
  IF PARAMCOUNT=0 THEN
  BEGIN
    WRITELN('Syntax : NomFichier[ [DestDriv]]');
    WRITELN;
    UNTILFLAG:=FALSE;
  END ELSE
  BEGIN
    UNTILFLAG:=TRUE;
    CMDLIN:='';
    FOR I:=1 TO PARAMCOUNT DO CMDLIN:=CMDLIN+PARAMSTR(I)+' '
  END;
  REPEAT
    IF NOT UNTILFLAG THEN
    REPEAT
      WRITE('*');
      READLN(CMDLIN)
    UNTIL CMDLIN<>'';
    FOR I:=1 TO LENGTH(CMDLIN) DO CMDLIN[I]:=UPCASE(CMDLIN[I]);
    I:=POS(' ',CMDLIN);
    IF I=0 THEN
    BEGIN
      NOM0:=CMDLIN;
      CMDLIN:=''
    END ELSE
    BEGIN
      NOM0:=COPY(CMDLIN,1,I-1);
      WHILE (CMDLIN[I]=' ') AND (I<LENGTH(CMDLIN)) DO I:=I+1;
      CMDLIN:=COPY(CMDLIN,I,2)
    END;
    OPENIN;
    IF IORESULT<>0 THEN WRITELN('File not found') ELSE
    BEGIN
      READCH(CH);
      READCH(CH1);
      IF (CH<>$76) OR (CH1<>$FF) THEN
      BEGIN
        WRITELN('Ce n''est pas un fichier compresse !')
      END ELSE
      BEGIN
        READCH(CH);
        CRC0:=CH;
        READCH(CH);
        CRC0:=CRC0 OR (CH SHL 8);
        NOM1:='';
        READCH(CH);
        WHILE CH<>0 DO
        BEGIN
          NOM1:=NOM1+CHR(CH);
          READCH(CH)
        END;
        NOM1:=CMDLIN+NOM1;
        WRITELN(NOM0,' ===> ',NOM1);
        READCH(CH);
        LENTAB:=CH;
        READCH(CH);
        LENTAB:=(LENTAB+CH*256)*4;
        IF LENTAB=0 THEN
        BEGIN
          WRITELN('ABANDON -- LENTAB=0!');
          HALT
        END;
        FOR I:=0 TO LENTAB-1 DO
        BEGIN
          READCH(CH);
          TABLE[I]:=CH
        END;
        OPENOUT;
        CRC:=0;
        WHILE GETCHAR(CH) DO
        BEGIN
          PUTCHAR(TAMPON^,CH);
          CRC:=CRC+CH
        END;
        IF CRC<>CRC0 THEN WRITELN('*** Attention: Erreur dans ChekSum ***');
        CLOSEOUT
      END
    END;
    CLOSE(FICH)
  UNTIL UNTILFLAG
END.
```

Les possesseurs de THOMSON TO9+ désireux d'exploiter au mieux les ressources de leur système trouveront ici les adresses essentielles à la manipulation des routines du moniteur. Ces routines sont utilisables depuis FORTH ou depuis une routine écrite en langage machine.

CONVENTION D'ECRITURE

Dans ce propos, les nombres ne figurant pas explicitement dans une routine FORTH seront suivis de la lettre d s'ils sont en décimal, de la lettre h s'ils sont en hexadécimal. Exemple:

10h = 16d

Les registres RAM sur deux octets seront symbolisés par el:XXXXh où XXXX représente l'élément mémoire d'adresse adr et adr+1. Exemple:

contenu de el:605Ah

soit en FORTH:

```
HEX 605A @ DECIMAL
```

Les nombres binaires seront suivis de la lettre b, b0 étant le bit de poids faible, b7 le bit de poids fort pour un octet, b15 le bit de poids fort d'un élément mémoire el:XXXX.

Les valeurs quelconques sur 8 bits sont nommées c et n sur 16 bits.

LE MONITEUR DU TO9+

INTERRUPTIONS:

TIMER ACIA clavier et souris IRQ FIRQ SWI	programmables

Le TIMER est initialisé à 100 ms et gère le clignotement du curseur. L'aiguillage sur ses propres fonctions est réalisé en mettant l'adresse de sa propre routine dans el:6027h (TIMEPT) correspondant à l'adresse de début d'exécution de votre sous-programme. Valider en mettant à 1 b5 de 6019h. Sauvegarder les registres DP et S en début de sous-programme. Terminer le sous-programme par JMP E830h (routine KBIN).

Pour les IRQ, mettre l'adresse de votre sous-programme dans el:6021 (IRQPT).

Pour les FIRQ, mettre l'adresse de votre sous-programme dans el:6023 (FIRQPT).

Pour les SWI, mettre l'adresse de votre sous-programme dans el:602F (SWI1). Les SWI2 et SWI3 sautent directement en 6800h et 7000h.

Lors de la mise sous tension, le moniteur effectue une réinitialisation des paramètres d'interruption et de diverses adresses mémoires. Les registres contenant les adresses du générateur de caractère standard et du générateur de caractères utilisateur sont restaurées avec les valeurs des tables standard.

Les interruptions sont invalidées. Le curseur devient invisible. La taille du RAM disk remise à 0 à la mise sous tension, reste inchangée lors d'un RESET. La banque RAM 1 est prise par défaut.

Le contenu de 6074h est réinitialisé (CONFIG). Le modem et les extensions situées entre E7F0h et E7F5h sont réinitialisées (décrochage de ligne dans le cas du MODEM en communication).

LE GENERATEUR DE CARACTERES

L'alphabet correspondant au standard ASCII est implanté dans le générateur G0. Chaque caractère est formé d'une suite de 8 octets, donnant en binaire:

adr+7	0 0 0 0 0 0 0 0	00h	matrice du "A"
adr+6	0 0 0 1 1 0 0 0	18h	
adr+5	0 0 1 0 0 1 0 0	24h	
adr+4	0 1 0 0 0 0 1 0	42h	
adr+3	0 1 1 1 1 1 1 0	7Eh	
adr+2	0 1 0 0 0 0 1 0	42h	
adr+1	0 1 0 0 0 0 1 0	42h	
adr	0 0 0 0 0 0 0 0	00h	

Cette table débute en RAM et son adresse est implantée dans el:60CFh (PTGENE). Si vous redéfinissez votre propre table, implantez l'adresse de votre nouvelle table dans PTGENE. Exemple:

```
CREATE TABLE NOUVEAUX-CAR  8 128 32 - * ALLOT
HEX 00 C, 00 C, etc...
puis
NOUVEAUX-CAR 60CF !  DECIMAL
```

L'alphabet G2 contient 22 caratères. Les accents, le c cédille faisant partie de cette table sont destinés à être combinés aux caratères de la table G0. La table G2 fait suite à la table G0 en mémoire.

La table utilisateur est implantée en mémoire vive et son adresse est déposée dans le registre el:602Dh (USERAF). Les codes "ASCII" sont pris séquentiellement à partir de 80h. On peut donc définir 128 caractères utilisateur.

LE MODE 40 COLONNES

Le TO9+ gère un écran 40 colonnes de 320x200 points avec 16 couleurs de forme, 16 couleurs de fond, 16 couleurs de tour. Chaque couleur est choisie parmis une palette de 4096 tons. Le mode compatible TO7 affiche 25 lignes 40 colonnes de caractères. Les caractères sont définis dans une matrice 8x8.

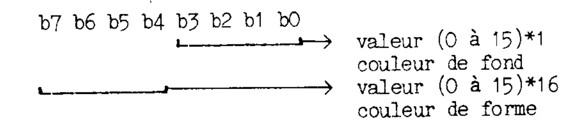
En mode graphique, l'écran est composé de 8000 octets de forme et 8000 octets de couleur. Chaque paire d'octet est située à la même adresse. On parlera alors de mémoire forme ou mémoire couleur, chaque mémoire étant sélectionnée par un bit du PIA (détail plus loin).

Pour chaque octet, un point peut prendre deux couleurs; une couleur forme si le point correspond à un bit à un; une couleur fond si le bit est à 0. La couleur est déterminée par le contenu de l'octet situé à la même adresse dans la mémoire couleur. Exemple:

adr	-->	c	0 0 1 1 1 1 1 1	3Fh en mem forme
adr	-->	c	1 1 1 1 0 0 0 0	0Fh en mem couleur

soit un segment graphique —XXXXXX avec deux points à gauche en noir, le reste en blanc.

La signification du contenu de l'octet couleur est la suivante:



avec

1000 NOIR	1001 ROUGE	1010 VERT
1011 JAUNE	1100 BLEU	1101 MAGENTA
1110 CYAN	1111 BLANC	
0000 GRIS	0001 ROSE	0010 VERT CLAIR
0011 SABLE	0100 BLEU CLR	0101 PARME
0110 BLEU CIEL	0111 ORANGE	

LE MODE 80 COLONNES

Dans ce mode, la définition passe à 640x200 points. Deux couleurs seulement sont disponibles pour la totalité de l'écran, une couleur de forme et une couleur de fond. A noter que l'affichage de ce mode sur un moniteur couleur ordinaire est assez rapidement fatiguante si l'on envisage de taper longtemps du texte.

La mémoire est organisée comme suit: la mémoire forme contient les bits des points des colonnes paires, la mémoire couleur celle des bits des points des colonnes

impaires. Exemple:

```
adr     c     0 0 1 1 1 1 1 1     en mem forme
adr     c     0 0 0 0 1 1 0 0     en mem couleur
```

donne à l'affichage

```
--XXXXXX----XX--
```

soit un segment de 16 points pour une ligne de 80 segments càd 640 points. Les bits à 0 prennent la couleur de fond, les bits à 1 la couleur de forme.

La mémoire écran débute en 4000h.

MODES PAGE 1, PAGE 2 ET OVERLAY

Ce mode affiche en 40 colonnes, mais les informations binaires sont organisées de manière différente. L'affichage utilise deux pages distinctes avec seulement 2 couleurs par page. Pendant l'affichage d'une page, l'autre peut êere remplie puis rendue visible par commutation. Dans le mode OVERLAY, ces deux pages peuvent être superposées. Les effets de ce dernier mode ne sont pas inintéressants.

La page 1 travaille dans la mémoire forme. Les bits à 1 prennent la couleur de forme, les bits à 0 la couleur de fond.

La page 2 travaille dans la mémoire couleur. Les bits à 1 prennent la couleur de forme, les bits à 0 la couleur de fond.

LE MODE BITMAP 16 COULEURS

Dans ce mode, chaque point peut prendre 16 couleurs à raison de 160x200 points. Pour chaque groupe de 4 points pn aura:

```
point 0:    b7 b6 b5 b4    de c mem forme
point 1:    b3 b2 b1 b0    de c mem forme
point 2:    b7 b6 b5 b4    de c mem couleur
point 3:    b3 b2 b1 b0    de c mem couleur
```

LE MODE TRIPLE OVERLAY

Ce mode superpose 4 pages avec des priorités d'affichage. La résolution est réduite à 160x200 points. Les priorités des plans vont de P1 à P4 dans l'ordre décroissant, P1 étant le plan le plus prioritaire. L'organisation mémoire est similaire au mode bitmap 16 couleurs:

```
plan 1:  b7 b6 b5 b4   de mem forme
plan 2:  b3 b2 b1 b0   de mem forme
plan 3:  b7 b6 b5 b4   de mem couleur
plan 4:  b3 b2 b1 b0   de mem couleur
```

Ces quatres plans ont la même couleur de fond. Les couleurs de forme sont pour P1 à l'arrière 1, 2, 4, 8.

En mode incrustation (à condition de disposer de l'interface d'incrustation) le TO9+ remplace la couleur noire par l'image vidéo. Ainsi, en affichant un écran compètement noir et une ou deux lignes en couleur en bas de page, on peut créer ses propres sous-titrages. En sélectionnant les options suivantes sous FORTH:

```
COLOROFF 1 SCROLL
```

on peut réaliser un banc titre à déroulement continu.

La sélection de la mémoire couleur est réalisée en mettant b0 de E7C3h à 0 ou en tapant COLORON sous FORTH.

La sélection de la mémoire forme est réalisée en mettant b0 de E7C3h à 1 ou en tapant COLOROFF sous FORTH.

GESTION DES CARACTERES

Le point d'entrée de la routine est E803h (PUTC). Le code du caractère à afficher doit être déposé dans le registre B:

entrée:

```
PUTC    registre 6809 B
```

Les codes des caractères normaux sont situés entre 20h et 7Fh. Les codes compris entre 07h et 1Fh contrôle certains paramètres de l'écran:

07h BEL émet un bip sonore

```
: BIP    7 EMIT  ;
```

08h	BS	déplace le curseur d'une case vers la gauche ou recopie le caractère courant si 6043h (COPCHR) contient FFh.
0Ah	LF	descend d'une ligne
0Bh	VT	remonte d'une ligne
0Ch	VT	efface le contenu de la fenètre courante
0Dh	CR	retour au début de la ligne courante
0Eh	SO	passage en mode TELETEL
0Fh	SI	retour en mode normal
10h	DLE	rien (utilisé en protocole TRANSPAC: ^P)
11h	DC1	allume le curseur et initialise TIMER
12h	DC2	répète le dernier caractère ASCII affiché exemple:

```
HEX : REPETE  ( n --) 12 EMIT EMIT ; DECIMAL
CR ." -" 39 REPETE
```

affiche une ligne de 40 "-".

13h	DC3	rien
14h	DC4	éteint le curseur
15h	NAK	rien
16h	ACC	pour séquence caractère
17h	ETB	rien
18h	CAN	efface le reste de la ligne courante à partir de la position courante du curseur.
19h 1Ah	EM SUB	rien
1Bh	ESC	séquence d'échappement (voir détail plus loin)
1Ch 1Dh	FS GS	rien
1Eh	RS	renvoi du curseur en ligne 0, colonne 0
1Fh	US	débute une séquence de positionnement du curseur

Affichage d'un caractère G2:

23h	livre sterling
24h	dollar
26h	dièse
27h	paragraphe
2Ch	flèche à gauche
2Dh	flèche en haut
2Eh	flèche à droite
2Fh	flèche vers le bas
30h	degré
31h	plus ou moins
38h	division entière
3Ch	1/4
3Dh	1/2
3Eh	3/4
6Ah	OE lié
7Ah	oe lié
7Bh	sz allemand

LES SEQUENCES D'ECHAPPEMENT

Ces séquences permettent de contrôler diverses fonctions:

48h	mode PAGE1
49h	mode PAGE2. Exemple:

```
HEX : PAGE£ ( n --)
      1B EMIT 1+ EMIT ; DECIMAL
```

```
0 PAGE£  passe en mode PAGE1
1 PAGE£  passe en mode PAGE2
```

4Ah	overlay PAGE2	
4Bh	overlay PAGE1	
4Ch	caractères taille normale !	
4Dh	caractères double hauteur !	voir le mot SIZE
4Eh	caractères double largeur !	en FORTH
4Fh	caractères double taille. !	
58h	masquage	
59h	mode bitmap 4 couleurs	
5Ah	40 colonnes	
5Bh	80 colonnes	

```
HEX : MODE ( n --)
      1B EMIT 1+ EMIT ; DECIMAL
```

0 MODE passe en 40 colonnes
1 MODE passe en 80 colonnes; à noter que l'éditeur FORTH fonctionne parfaitement dans ce mode.

5Ch	inversion vidéo. Voir le mot INVCOLOR en FORTH.
5Eh	mode bitmap 16 couleurs
5Fh	démasquage
68h	écrit caractère c sans changer la couleur
69h	écrit car c dans la couleur courante
6Ah	scroll à vitesse normale
6Bh	mode page
6Ch	suppression incrustation
6Dh	passage en mode incrustation
6Eh	scroll doux. Idéal pour les bancs titres en mode incrustation. Exemple:

0 BORDER 0 PAPER COLOR 7 INK COLOR CLS
1 SCROLL suivi de vos affichages, ceci en mode incrustation.

88h	mode triple overlay sélection page 1
89h	mode triple overlay sélection page 2
8Ah	mode triple overlay sélection page 3
8Bh	mode triple overlay sélection page 4

CLAVIER

La lecture rapide du clavier est exécutée par la routine située en E809h (KTST). Paramètre retouné par le registre 6809 CC.

Le décodage est effectué par la routine E806h (GETC). Paramètres d'entrée:

el:6079h BUFCLV
607Bh SIZCLV

et renvoie en retour:

registres 6809 B et CC

La programmation du clavier est effectuée par E806h (GETC) avec en entrée:

registre 6809 B
registre 6019h (STATUS)

les codes à affecter à STATUS sont:

F8h	réinitialise le clavier
F9h	CAPS LOOK on
FAh	CAPS LOOP off
FBh	sélectionne les codes spéciaux du keypad
FCh	sélectionne les chiffres pour le keypad
FDh	interrogation clavier: réponse B0h=CAPS ON; B4h=CAPS OFF

b1 de STATUS est remis à 0 après exécution de GETC.

GRAPHISME

L'allumage et extinction d'un point est réalisé par appel de la routine E80Fh (PLCT) avec en entrée:

registres 6809 X et Y
registres 6038h FORME
6041h CHDRAW
603Bh COLOUR
6019h STATUS

et en sortie:

registres el:603Dh PLOTX ! coordonnées du dernier
el:603Fh PLOTY ! point tracé.

Cette routine est compatible avec tous les modes graphiques. Il faut simplement veiller à ne pas sortir des limites graphiques du mode courant (160;320;640x200).

Le contenu de 6041h (CHDRAW) doit être mis à zéro, sinon on est en mode caractère.

Le tracé d'un segment de droite sera exécuté par appel de E80Ch (DRAW) avec en entrée

registres 6809 X et Y
registres el:603Dh PLOTX
el:603Fh PLOTY
6041h CHDRAW
6038h FORME
603Bh COLOUR
6019h STATUS

et en sortie:

registres el:603Dh PLOTX
el:603Fh PLOTY

La routine trace un segment de droite entre le point d'abscisse PLOTX et d'ordonnée PLOTY et les coordonnées X et Y passées par X et Y. Les coordonnées du dernier point allumé sont mises dans PLOTX et PLOTY.

Pour les segments horizontaux, la routine de tracé fait appel à une sous-routine de tracé rapide.

On peut accéder à la couleur d'un point en faisant appel à E821h (GETP) avec en entrée:

registres 6809 X et Y

et en sortie

registre 6809 B

Les différentes valeurs prises par le paramètre de retour dépendent du mode d'affichage:

- mode TO7: -16<=B<=-1 en couleur de fond et 0<=B<=15 en couleur de forme.
- mode bitmap 4 couleurs: 0<=B<=3
- mode bitmap 16 couleurs: 0<=B<=15
- modes PAGE1, PAGE2, OVERLAY, 80 colonnes, triple overlay: 0 pour couleur forme, -1 pour couleur fond.

LECTURE DES CARACTERES AFFICHES A L'ECRAN

Cette routine renvoie le code du caractère situé aux coordonnées texte par appel de E824h (GETC) avec en entrée:

registres 6809 A et X

et en sortie

registre 6809 B

X contient le numéro de colonne [1..80] et A le numéro de ligne [0..24]. Si le caractère n'est pas reconnu, B renvoie une valeur nulle. Cette routine n'est accessible que dans les modes TO7, 80 colonnes, PAGE1, PAGE2, overlay.

GENERATEUR MUSICAL

La note à jouer est passée par le registre B puis appel dela routine E81Eh (NOTE). On a en entrée:

registre 6809 B
registres el:6036h OCTAVE
el:6033h DUREE
el:6031h TEMPO
6035h TIMBRE

Les treize notes de base sont:

silence	30h	DO	31h
DO dièze	32h	RE	33h
RE dièze	34h	MI	35h
FA	36h	FA dièze	37h
SOL	38h	SOL dièze	39h
LA	3Ah	LA dièze	3Bh
SI	3Ch	DO+	3Dh

L'octave est comprise dans l'intervalle [1..5], avec les valeurs 16, 8, 4, 2 et 1 à affecter respectivement au registre OCTAVE.

La durée est controlée par le registre DUREE, de 96 à 0. 96 correspond à une ronde, 3 à une triple croche.

Le timbre est chargé avec une valeur allant de 0 à 5 dans le registre TIMBRE.

LE JOYSTICK

La position de la manette de jeu est délivrée par la routine E827h (JOYS) avec en entrée:

registre 6809 A

et en sortie:

registres 6809 B et CC.

On passe en entrée le numéro du joystick dont on désire connaître l'état. En retour, on aura la signification suivante dans le registre B:

0	rien	1	midi
2	1h30	3	3h
4	4h30	5	6h
6	7h30	7	9h
8	10h30		

CC=1 signifie bouton enfoncé.

L'INTERFACE RS232

Cet interface, disponible en option, permet de recevoir et de transmettre des données et des fichiers depuis ou vers une imprimante série, un modem, un autre ordinateur, ou tout autre appareil transmettant en série selon la norme RS232. Aucun logiciel de gestion de protocole n'est implanté dans le système THOMSON. La routine est située en E812h (RSCO) et accepte en entrée:

registre 6809 B	
registres 602Bh	RS.OPC
el:6044h	BAUDS
6046h	NOMBRE

et en sortie:

registre 6809 CC	
registre 602Ch	RS.STA

Les opérations sont sélectionnées en fonction du contenu affecté à RS.OPC:

0000 0001	ouverture en lecture/écriture série
0000 0010	lecture d'un caratère série
0000 0100	ouverture en écriture série seulement
0000 1000	écriture en //
0000 1001	écriture caractère en série
0000 1100	écriture caractère en série
0001 0000	fermeture en //
0001 0100	fermeture en série
0010 0000	copie graphique de l'écran
0100 0000	ouverture en écriture //

Le registre 6809 B contient le code du caractère à envoyer.

De plus, en transmission série, le contenu du registre NOMBRE permet la sélection de certaines options:

b7 b6	11	5 bits
	10	8 bits
	01	7 bits
	00	6 bits
b5	0	horloge système
	1	horloge externe
b4 b3 b2	111	bit d'espace
	101	bit de marque
	011	parité paire
	001	parité impaire
	000	pas de parité
b1	0	mode modem
	1	mode terminal
b0	0	2 bits de stop
	1	1 bit de stop

Quand à la vitesse de transmission, la plage d'utilisation est plus étendue que celle du THOMSON TO7-TO7/70. Cette vitesse est sélectionnée par la valeur affectée à BAUDS:

bauds	valeur	bauds	valeur
50	1	75	2
110	3	135	4
150	5	300	6
600	7	1200	8
1800	9	2400	Ah
3600	Bh	4800	Ch
7200	Dh	9600	Eh
19200	Fh		

LA TABLE D'ALLOCATION DES FICHIERS

Lors des accès disque, la FAT (File Allocation Table= Table d'Allocation des Fichiers) permet de prendre connaissance de l'organisation du disque courant:

octet 0:	0
octet 1:	bloc 0, piste 0, secteurs 1 à 8
octet 2:	bloc 1, piste 0, secteurs 9 à 16
etc...	
octet 160:	bloc 159, piste 79, secteurs 9 à 16

ceci en double densité. En simple densité, la FAT est limitée à 127 blocs.

Le contenu des octets de la FAT signifie:

FFh bloc non alloué
FEh bloc réservé
00..BFh bloc alloué, c indique le numéro du bloc logique suivant. Exemple:

octet 2 contient 3Fh, donc le bloc logique suivant est l bloc 3Fh.

C1h..C8h dernier bloc d'un fichier. Les bits b0 à b3 donnent le nombre de secteurs utilisés par le fichier dans ce dernier bloc.

La directory du disque a une capacité maximale de 112 fiches en double densité, 56 fiches en simple densité. Le nom de chaque fichier occupe 32 octets:

oct 00..07 nom du fichier
oct 08..0Ah extension du fichier: BAS BIN FTH BLK, etc..
oct 0Bh type du fichier:
0 programme BASIC ASCII ou binaire
1 données BASIC en ASCII
2 programme en langage machine
3 fichier assembleur en ASCII
oct 0Ch drapeau booléen: FFh=ASCII; 00=binaire
oct 0Dh numéro du premier bloc logique du fichier
oct 0Eh..0Fh nombre octets utiles du dernier secteur
oct 10h..17h commentaire associé au fichier
oct 18h..1Ah date fichier
oct 1Bh..1Fh réservés

Pour chaque entrée dans le catalogue, le premier octet indique l'état de cette entrée:

00h	pas de fichier.
20h à 7Fh	code ASCII du premier car du nom de fichier
FFh	fin du catalogue.

COMMUTATION DE BANQUES ROM

Les différentes banques mémoires situées entre les adresses 0000h à 3FFFh peuvent être sélectionnées par exécution de la routine EC03h (COM), avec en entrée:

registres 6809 U et A

Le registre U définit l'adresse de la routine à exécuter. Le registre A définit une valeur au format 00SS00BB, SS étant le numéro de SLOT et BB le numéro de banque. Les différentes valeurs possibles sont les suivantes:

—SS—BB
11 xx Cartouche externe
— 00 Basic 128
— 01 Extramoniteur
— 10 Basic 1
— 11 Exploitation des fichiers

— définit un état différent de 1. L'adresse 20h de chaque slot donne le nombre de banques ROM, ceci dans le cas où le programme situé en ROM tient sur plus de 16K.

Les appels à l'extra-moniteur sont effectués à l'aide de la routine ECOCh (EXTRA), avec en entrée:

registres 6809 B
registres pages 6100h à 62FFh, dépendant de la routine appelée.

Le registre B contient le numéro de la routine à exécuter.

LA CARTE MEMOIRE

adresses hexa
0000-3FFF espace ROM
4000-5FFF 2x8k mémoire écran
6000-60FF registres du moniteur
6100-9FFF ram utilisateur banque fixe
A000-DFFF 14 banques RAM 16k en parrallèle
E000-E7AF 2x1.9k rom pour le disque
E7B0-E7BF 16 adresses non utilisées ni décodées
E7C0-E7C7 PIA 6846 système
E7C8-E7CB PIA 6821 système
E7CC-E7CF PIA 6821 extension jeux
E7D0-E7D7 contrôleur de disque
E7D8-E7D9 sélection allouée au floppy
E7DA-E7DB palette
E7DC-E7DD gate Mode Page en affichage
E7DE-E7DF 6850 pour liaison clavier
E7E0-E7E3 interface de communication TO7-70
E7E4-E7E7 compteurs crayon optique
E7E8-E7EB interface RS232 externe
E7F0-E7F7 interface IEEE externe
E7F8-E7FD modem
E7FE-E7FF réservés
E800-FFFF 2x6k du moniteur.

La commutation des banques mémoires RAM peut être réalisée par un programme du type:

```
COMMUT  PSHS D,X,U
        LDU  =$E7C0H
        LDB  11,U
        ANDB =$0FBH
        STB  11,U
        LDX  =TAB table des valeurs à mettre dans PIA

        LDA  A,X
        STA  9,U  modifie les directions du PIA

        ORB  =04
        STB  11,U
        PULS D,X,U,PC

TAB     FCB  $0FH,$17H,$E7H,$67H,$$A7H,$27H
```

Ce programme est compatible TO7/70. Pour commuter les autres banques, il faut programmer le Gate Mode Page. Le nombre maximal de banques possible est 32 (14 implantées), soit une capacité mémoire potentielle de 32x16k, soit 512k auquels viennent s'ajouter 16k de mémoire vive non commutable. Avec la cartouche FORTH, la commutation des banques est réalisée par la commande n BANK, où n correspond au numéro de banque sélectionnée. La partie mémoire située dans les quatres derniers K de la banque courante sont réservés aux tampons d'édition. ■

Q+R

Voici une nouvelle série de questions posées par nos adhérents. Les réponses à ces questions paraîtront dès que vous nous les enverrez. Si vous mêmes avez des questions, vous pouvez nous les adresser. Jouez le jeu, si vous avez une réponse à une question, même partiellement, écrivez.

QUESTION 3: de A.JACCOMARD (29190 PLEYBEN)
L'article règle à calcul dans JEDI no 20, janvier 86 donne pour 1/K et 1/K' des développements ne correspondant pas aux résultats annoncés (même après interversion de ceux-ci), et ils sont indispensables pour améliorer la précision. Quelqu'un a-t-il les valeurs exactes?

QUESTION 4: de Michel ZUPAN (67800 HOENHEIM)
RECURSIVITE: comment définir RECURSE ou MYSELF ou un autre mot pour qu'il prévienne un overflow de la pile de retour?

QUESTION 5: de Michel ZUPAN (67800 HOENHEIM)
AUTO-EDITION: comment vectoriser EMIT pour qu'il écrive dans les écrans? Application: ré-écrire facilement le source d'un mot-code avec virgule et C-virgule sans plus passer par ASSEMBLER.

QUESTION 6: de Michel ZUPAN (67800 HOENHEIM)
COMPILATION CONDITIONNELLE: comment n'autoriser la compilation d'un mot que s'il ne figure pas déjà dans le CURRENT? Définir une variable booléenne REDEF? autorisant ou interdisant la définition de doublons dans un vocabulaire. En cas d'interdiction, la définition doit être sautée sans ABORT du système.

QUESTION 7: de Michel ZUPAN (67800 HOENHEIM)
SMART-INTERPRETER: comment changer l'interpréteur d'exécution pour qu'il agisse ainsi:
- si le mot existe je l'exécute.
- si le mot n'existe pas, j'en crée un.
Application: FORTHLOG en F83...

QUESTION 8: de Michel ZUPAN (67800 HOENHEIM)
CROSS-INTERPRETER: comment compiler des fichiers texte d'origine quelconque (WORDSTAR ou autre)?

QUESTION 9: de Michel ZUPAN (67800 HOENHEIM)
TRANSLATOR: comment traduire un fichier FORTH en fichier texte quelconque (WORDSTAR ou autre: vous m'avez compris)?
Ndlr: voir une solution proposée par A.JACCOMARD dans ce même numéro. ■

ARITHMETIQUE DES ENTIERS EQUILIBRES.

par Vic NORTON,

Département de Mathémathiques et de Statistiques,

Université de BOWLING GREEN, OHIO.

> "L'Esprit Divin trouva une issue sublime en cette merveille de l'analyse, le présage de l'idéal, le milieu entre être et non-être, que nous nommons la racine [carrée] imaginaire de l'unité négative."
>
> - Gottfried Wilhelm Leibniz

> "Dieu fit les entiers, tout le reste est travail d'Homme."
>
> - Leopold Kronecker

1 - Les entiers numéraux.

L'arithmétique sur les entiers m'intéresse. Mais qu'est-ce qu'un entier (vous ne l'avez certainement pas demandé) ? Comment représente-t-on un entier ?

Supposons, pour le moment, que nous savons tous ce qu'est un entier, et aussi que nous savons additionner et multiplier des entiers. Etant donnés une base β et un ensemble fini de chiffres entiers D, on pose

$$[a_n \dots a_1 a_0] = a_n\beta^n + \dots + a_1\beta + a_0 ,$$

où $a_i \in D$ pour $i = 0,1,\dots,n$. Avec certaines hypothèses sur β et D, ceci représente la notation de position usuelle pour un entier (sauf que les crochets carrés, les exposants et les indices sont omis). Les humains prennent $\beta = 10$ et $D = \{0,1,\dots,9\}$; mais les calculateurs numériques modernes, n'ayant que deux doigts, préfèrent $\beta = 2$ et $D = \{0,1\}$.

2 - Arithmétique positionnelle.

Du point de vue "calculs", un numéral entier est un tableau de chiffres

$$a = a[0],a[1],a[2],\dots$$

Les algorithmes suivants sont utilisés pour l'addition et la multiplication des numéraux :

```
{ pour additionner :  c:= a + b }
   c:= 0 ;  carry:= 0 ;
   n:= max (long(a),long(b));
   FOR i:=0  TO  n-1  DO
 c[i]:=a[i] + b[i] ;
 c[n] <- carry
{ pour multiplier :  c:= a * b }
   c:= 0;  carry:= 0;
   FOR i:= 0  TO  long(a)-1  DO
   BEGIN
   FOR j:= 0  TO  long(b)-1  DO
   c[i + j]:= c[i + j] + (a[i] * b[j]);
   c[i + long(b)] <- carry
   END
```

Quelques explications sont nécessaires.

Tout d'abord, la longueur d'un numéral est nulle si tous ses chiffres sont zéro. Autrement long(a) = n si a[n-1] <> 0 et a[i] = 0 pour i ≥ n.

Ensuite, "carry" représente un emplacement réservé dans la mémoire de la machine pouvant contenir les "retenues" résultant de l'addition de deux nombres et d'une retenue.

Troisièmement,, l'opération + ajoute deux chiffres et le contenu courant de "carry", retournant un chiffre et ajustant "carry" pour l'opération suivante. L'opération * multiplie deux chiffres et retourne le produit. Aucune retenue n'est associée à la multiplication de chiffres dans les systèmes que je vais considérer.

Enfin, l'instruction "c[n] <- carry" de l'algorithme d'addition transfère le contenu courant de carry vers le numéral c, en commençant à l'emplacement c[n], et laisse un carry nul. De même "c[i + long(b)] <- carry" de l'algorithme de multiplication transfère le carry vers c à la position i + long(b).

3 - Le système ternaire équilibré.

Considérons la base $\beta = 3$ et l'ensemble d'entiers $D = \{0,\pm 1\}$. Tout entier z non nul peut être représenté par la forme unique

$$z = [a_n \dots a_1 a_0]_\beta$$

avec $a_i \in D$ pour $i = 0,1,\dots,n$ et a_n <> 0. Ceci est appelé le "système ternaire équilibré".

Une jolie caractéristique de ce système est l'absence de bit de signe. Tous les entiers peuvent être représentés par des numéraux. Les entiers positifs commencent par un chiffre 1, les entiers négatifs par

-1. Si on fait $\bar{a} = -a$, la négation est obtenue de façon triviale par

$$-(a_n \dots a_1 a_0)_3 = (\bar{a}_n \dots \bar{a}_1 \bar{a}_0)_3$$

Pour simplifier la représentation numérale, j'adopte le symbole 2 pour le chiffre -1. Mon ensemble de chiffres est alors 0, 1, 2 avec la table de multiplication

$$0 * a = 0, \quad 1 * a = a, \quad 2 * 2 = 1.$$

La Table 1 montre l'addition de deux chiffres et d'une retenue. L'algorithme de la section 2 peut être utilisé pour ajouter ou multiplier des numéraux arbitraires.

TABLE I : addition ternaire.

SOMMECARRY

	0	1	2
0 + 0	00	01	02
0 + 1	01	12	00
0 + 2	02	00	21
1 + 1	12	10	01
1 + 2	00	01	02
2 + 2	21	02	20

Pour concrétiser ces idées, considérons

$$\begin{array}{r} 886 = 3^6 + 3^5 - 3^4 - 3^2 + 3 + 1 \\ + \; 592 = 3^6 - 3^5 + 3^4 + 3^3 - 3 + 1 \\ \hline 1478 = 3^7 - 3^6 + 3^3 - 3^2 + 3 - 1 \end{array}$$

Utilisez la Table 1 pour calculer

$$\begin{array}{r} 1120211 \\ + \; 1211021 \\ \hline 12001212 \end{array}$$

De même :

$$\begin{array}{r} 277 = 3^5 + 3^3 + 3^2 - 3 + 1 \\ \times \; -2 = \qquad\qquad - 3 + 1 \\ \hline -554 = -3^6 + 3^5 - 3^4 + 3^2 + 1 \end{array}$$

est calculée comme

$$\begin{array}{r} 101121 \\ \times \quad 21 \\ \hline 101121 \\ 202212 \\ \hline 2120111 \end{array}$$

Voici les 18 premiers numéraux ternaires positifs :

1	2	3	4	5	6
1	12	10	11	122	120

7	8	9	10	11	12
121	102	100	101	112	110

13	14	15	16	17	18
111	1222	1220	1221	1202	1200

4 - Entiers.

J'ai décrit le système ternaire équilibré, non pas parce que je désire faire des calculs dans ce système, mais parce qu'il ressemble aux systèmes dans lesquels je veux faire des calculs. Mais avant de parler de ces derniers, je dois revenir sur la question "qu'est-ce qu'un entier ?".

Ce que j'ai appelé jusqu'ici "entiers" sont en fait des entiers rationnels : racines rationnelles de polynômes unitaires à coefficients entiers. En particulier, tout entier rationnel m résoud l'équation $x - m = 0$.

Un entier algébrique est une extension naturelle de cette idée. Il est racine (peut-être complexe) d'un polynôme unitaire à coefficients entiers. Les nombres

$$\sqrt{2}, \quad (1 + \sqrt{5})/2, \quad \sqrt{-1}, \quad (1 + \sqrt{-3})/2,$$

sont des entiers algébriques de degré 2. Ils sont racines respectivement des polynômes

$$X^2 - 2, \quad X^2 - X - 1, \quad X^2 + 1, \quad X^2 - X + 1$$

Le nombre $(1 + \sqrt{3})/2$ n'est pas un entier algébrique. Il résoud $2X^2 - 2X - 1 = 0$, mais celle-ci n'est pas une équation unitaire à coefficients entiers.

Soit Ω l'un des entiers algébriques de degré 2 ci-dessus. Alors pour toute paire d'entiers (rationnels) m et n, le nombre $z = m + n\Omega$ est aussi un entier algébrique.

Pour le montrer, prenons par exemple

$$\Omega = (1 + \sqrt{-3})/2 \quad \text{et} \quad z = 3 + 5\Omega$$

Alors

$$\begin{aligned} 0 &= 25(\Omega^2 - \Omega + 1) \\ &= 25[(z - 3)^2/5^2 - (z - 3)/5 + 1] \\ &= z^2 - 11z + 49 \end{aligned}$$

montre que z est racine de l'équation unitaire

$$X^2 - 11X + 49 = 0$$

Les entiers algébriques de la forme $m + n\Omega$ (où Ω est l'un des nombres ci-dessus) peuvent être additionnés, soustraits, ou multipliés pour produire des entiers algébriques de même forme. Soit encore $\Omega = (1 + \sqrt{-3})/2$; alors

```
  3 + 5Ω          3 + 5Ω
+ 2 - 7Ω        x 2 - 7 Ω
--------       ---------
  5 - 2Ω       6 - 11Ω - 35Ω²
             = 6 - 11Ω - 35 (Ω-1)
             = 41 - 46Ω
```

Les quatre ensembles d'entiers algébriques de forme m + n sont clos sous l'addition, la soustraction et la multiplication. Ils possèdent aussi un algorithme de division, comme celui des entiers rationnels : étant donnés 2 entiers a, b avec b <> 0 il existe des entiers

q, r tels que $a = bq + r$ et $N(r) < N(b)$.

Pour des entiers rationnels, $N(a) = a^2$. Pour des entiers algébriques de degré 2, $N(a) = |a\bar{a}|$, où $\bar{a}$ est obtenu à partir de l'expression étendue de a en remplaçant les racines carrées par leurs négations.

J'ai inclus dans cette discussion les réels algébriques entiers $\sqrt{2}$ et $(1 + \sqrt{5})/2$ uniquement à titre d'illustration. Je remarquerais en passant que l'arithmétique de ces entiers réels est bien plus difficile que celle basée sur les entiers complexes $\sqrt{-1}$ et $(1 + \sqrt{-3})/2$.

J'appelle les entiers algébriques de la forme m + nΩ, avec $\Omega = \sqrt{-1}$ ou $(1 + \sqrt{-3})/2$ des entiers de Gauss, bien que cette dénomination soit habituellement réservée au cas $\Omega = \sqrt{-1}$. Je vais faire maintenant de l'arithmétique positionnelle sur des entiers de Gauss!

5 - Systèmes quinaire et septénaire équilibrés.

Les chiffres non nuls du système ternaire équilibré sont les racines carrées de l'unité : les racines de l'équation $X^2 = 1$. Ils sont les puissances de $\Omega = -1$. Les systèmes numériques équilibrés pour les entiers de Gauss sont obtenus d'une façon analogue, leurs chiffres non nuls étant les racines 4ème et 6ème de l'unité, les puissances de $\Omega = \sqrt{-1}$ et $\Omega = (1 + \sqrt{-3})/2$ respectivement. Je nomme ces systèmes de nombres les systèmes quinaire et septenaire équilibrés.

Une représentation géométrique des chiffres de ces 3 systèmes équilibrés est montrée en Figure 1. Par facilité j'utilise les symboles 0, 1, 2, 3, 4, 5, 6, avec Ω noté 2 dans tous les cas.

Figure 1

```
                    B.  2   .A        3.___.2
                       / \            /     \
2___.___1          3. ( 0. ) .1    4. ( 0. ) .1
                       \ /            \     /
                    C'  4.   'D       5.___.6
```

Ternaire équilibré $\Omega = -1$	Quinaire équilibré $\Omega = \sqrt{-1}$	Septénaire équilibré $\Omega = (1 + \sqrt{-3})/2$
0 = 0	0 = 0	0 = 0
1 = 1	1 = 1	1 = 1
2 = Ω	2 = Ω	2 = Ω
	$3 = \Omega^2 = -1$	$3 = \Omega^2 = -1 + \Omega$
	$4 = \Omega^3 = -\Omega$	$4 = \Omega^3 = -1$
		$5 = \Omega^4 = -\Omega$
		$6 = \Omega^5 = 1 - \Omega$

Quatre nombres autres que les chiffres jouent un rôle particulier dans le système quinaire. Ce sont les "retenues", des "non-chiffres", que j'ai étiquetées A, B, C, D, en Fig. 1. Ils sont en fait représentés par 13, 24, 31 et 42 respectivement.

Soit $\Omega = \sqrt{-1}$ ou $\Omega = (1 + \sqrt{-3})/2$. Posons

$\beta = 2 + \Omega$ et

$D = \{0, 1, \Omega, \Omega^2, \Omega^3, \Omega^4, \Omega^5\}$

(Ω^4 et Ω^5 sont redondants pour $\Omega = \sqrt{-1}$). Alors tout entier gaussien z non nul peut être représenté de façon unique sous la forme

$z = [a_n \ldots a_1 a_0]_\beta$

avec $a_i \in D$ pour $i = 0, 1, \ldots, n$ et $a_n <> 0$

Si $\Omega = \sqrt{-1}$, cette représentation définit le système quinaire équilibré ; si $\Omega = (1 + \sqrt{-3})/2$, le système septénaire équilibré.

Je représente par p le nombre de chiffres non nuls dans D, pour chacun des trois systèmes. Ainsi p = 2 pour le système ternaire, 4 pour le quinaire et 6 pour le septénaire. Avec les conventions de la fig. 1,

$D = \{0, 1, \ldots, p\}$

La multiplication de chiffres pour chacun des trois systèmes équilibrés est simplement décrite par

$0 * a = 0$ pour tout $a \in D$,

$a * b = 1 + ((a + b - 1) \bmod p)$

pour tout $a, b \in D - \{0\}$

Ici l'opération r mod p doit retourner un entier du domaine 0,1,...,p-1.

La négation d'un nombre s'obtient par

$\bar{a} = 1 + ((1 - a) \bmod p)$,

et comme pour tous les systèmes équilibrés

$-a_n \ldots a_1 a_0 = \bar{a}_n \ldots \bar{a}_1 \bar{a}_0$.

La Table 2 représente l'addition de deux chiffres et d'une retenue dans un système quinaire équilibré, et la Table 3 celle d'un système septénaire. Le système quinaire se distingue parmi les trois systèmes

équilibrés par le fait que les retenues ne sont pas toujours des chiffres. L'algorithme de la section 2, avec la Table 2 ou 3, peut être utilisé pour additionner ou multiplier des numéraux quinaires ou septénaires quelconques.

TABLE 2 : addition quinaire.

SOMMES	RETENUES								
	0	1	2	3	4	A	B	C	D
0+0	00	01	02	03	04	13	24	31	42
0+1	01	14	13	00	42	10	02	04	D3
0+2	02	13	21	24	00	A4	20	03	01
0+3	03	00	24	32	31	02	B1	30	04
0+4	04	42	00	31	43	01	03	C2	40
1+1	14	D2	10	01	D3	11	13	42	D0
1+2	13	10	A4	02	01	12	21	00	14
1+3	00	01	02	03	04	13	24	31	42
1+4	42	D3	01	04	40	14	00	43	41
2+2	21	A4	A3	20	02	A0	22	24	13
2+3	24	02	20	B1	03	21	23	32	00
2+4	00	01	02	03	04	13	24	31	42
3+3	32	03	B1	B4	30	24	B0	33	31
3+4	31	04	03	30	C2	00	32	34	43
4+4	43	40	04	C2	C1	42	31	C0	44

TABLE 3 : addition septénaire.

SOMMES	RETENUES						
	0	1	2	3	4	5	6
0+0	00	01	02	03	04	05	06
0+1	01	15	14	02	00	06	63
0+2	02	14	26	25	03	00	01
0+3	03	02	25	31	36	04	00
0+4	04	00	03	36	42	41	05
0+5	05	06	00	04	41	53	52
0+6	06	63	01	00	05	52	64
1+1	15	16	10	14	01	63	62
1+2	14	10	13	26	02	01	15
1+3	02	14	26	25	03	00	01
1+4	00	01	02	03	04	05	06
1+5	06	63	01	00	05	52	64
1+6	63	62	15	01	06	64	60
2+2	26	13	21	20	25	02	14
2+3	25	26	20	24	31	03	02
2+4	03	02	25	31	36	04	00
2+5	00	01	02	03	04	05	06
2+6	01	15	14	02	00	06	63
3+3	31	25	24	32	30	36	03
3+4	36	03	31	30	35	42	04
3+5	04	00	03	36	42	41	05
3+6	00	01	02	03	04	05	06
4+4	42	04	36	35	43	40	41
4+5	41	05	04	42	40	46	53
4+6	05	06	00	04	41	53	52
5+5	53	52	05	41	46	54	50
5+6	52	64	06	05	53	50	51
6+6	64	60	63	06	52	51	65

Voici des exemples d'addition et de multiplication dans le système quinaire équilibré, les retenues sont expressément indiquées pour l'addition.

```
  A2C4          12
  ----        x 34
  1234          41
+ 2241          34
134422        3131
```

Sous forme gaussienne plus usuelle, ces mêmes calculs se présentent ainsi :

```
   -4 + 12Ω          2 + 2Ω
+ -13 +  3Ω       x -2 - 2Ω
  -17 + 15Ω       -4 -8Ω - 4Ω²
                = -4 - 8Ω - 4(-1)
                = -8Ω
```

Vous pouvez vérifier que
$3131 \equiv -(2 + \Omega)^3 + (2 + \Omega)^2 - (2 + \Omega) + 1$
$= -8\Omega$ avec $\Omega^2 = -1$.

6 - Implantation en FORTH.

J'ai implanté l'arithmétique, autre que la division, pour les trois systèmes équilibrés. La division reste un problème, que je discute dans la section suivante.

La multiplication (ou la division) de chiffres dans un système équilibré est accomplie de façon triviale par l'arithmétique modulaire indiquée en section 5. Pour implanter les algorithmes généraux d'addition et de soustraction de la section 2, la seule difficulté provient de l'addition de deux chiffres et une retenue.

Considérons l'une des tables 1, 2 ou 3 d'addition équilibrée. Chaque entrée interne consiste en une retenue à gauche et un chiffre à droite. Ainsi le contenu de la table peut être mémorisé dans deux tableaux rectangulaires, une table des chiffres et une table des retenues, chacune étant indexée par les sommes de chiffres et de retenues. Une fois les tables placées en mémoire, l'addition de deux chiffres et d'une retenue peut être exécutée par le calculateur comme je l'ai fait à la main dans les exemples.

Les tables d'addition conviennent au calculateur humain. Mais à cause de la circularité de la multiplication de chiffres, beaucoup de leur contenu est redondant. Par exemple, considérons l'entrée A0 de la ligne 2 + 2, colonne A de la Table 2. Divisons par 2 les indices de ligne et de colonne (déplacez chaque chiffre en arrière de 1):

(2 + 2)/2 = 1 + 1, A/2 = 13/2 = 42 = D.

Sur la ligne 1 + 1, on lit en colonne D l'entrée D0. Multiplions alors D0 par 2 (déplacez chaque chiffre non nul en avant de 1):

2 * D0 = 2 * 420 = 130 = A0,

et nous retrouvons l'entrée A0 de la ligne 2 + 2, colonne A.

Dans mon implantation FORTH, j'ai placé les tables des chiffres et retenues correspondant aux Tables 1, 2 et 3 dans des tableaux d'octets de dimensions 2 x 2, 3 x 9, et 4 x 7 respectivement. La dimension de la colonne dans chaque cas est le nombre de retenues requises.

Pour éviter la confusion dûe aux détails superflus, je voudrais éviter autant que possible la distinction entre les trois systèmes. Les seules différences importantes se trouvent dans les nombres p = 2, 4, 6, de la division modulo mentionnés en section 5, et dans les tables d'addition

J'ai placé p dans une variable nommée PBASE. Pour sélectionner un système (p et tables), j'exécute TRIT, QUINT, ou SEPT avec les définitions FORTH:

```
: TRIT ( -- ) 2 PBASE !
    ' TDIGITS CFA 'DT !
    ' TCARRIES CFA 'CT ! ;
: QUINT ( -- ) 4 PBASE !
    ' QDIGITS CFA 'DT !
    ' QCARRIES CFA 'CT ! ;
: SEPT ( -- ) 6 PBASE !
    ' SDIGITS CFA 'DT !
    ' SCARRIES CFA 'CT ! ;
```

Ainsi les tables d'addition

TDIGITS, TCARRIES, QDIGITS, QCARRIES, SDIGITS, SCARRIES

sont vectorisées sur les variables 'DT et 'CT. Ces variables à leur tour sont lues par les mots

```
: DIGITS-TABLE ( ligne colonne -- chiffre )
    'DT @ EXECUTE ;
: CARRIES-TABLE ( ligne colonne -- retenue )
    'CT @ EXECUTE ;
```

Les algorithmes généraux d'addition et de multiplication de la section 2 sont exécutés par les mots Z+ et Z*. J'en dirai plus dans un moment, lorsque je parlerai de mon système pseudo-FORTH pour les numéraux équilibrés. Je voudrais faire remarquer ici que ces deux mots, Z+ et Z* sont définis à partir des mots PBASE, DIGITS-TABLE et CARRIES-TABLE, eux-mêmes prenant leur sens de TRIT, QUINT ou SEPT, dépendant du système désiré.

* * *

Tous à cette conférence sommes des amoureux de FORTH. Sinon pourquoi serions-nous là? Charles MOORE a développé FORTH pour traiter des entiers (entre autres choses) avec la conviction qu'ils sont suffisants en pratique. Bien que mon but ne puisse être considéré comme "pratique" que par les esprits les plus éclairés, le mieux pour manipuler ma notion étendue d'entiers n'est-il pas de copier le système de Moore?

L'ingrédient fondamental en FORTH, c'est la pile de données. J'ai appelé mes entiers des "z-numéraux". Pour les manipuler (chaînes d'octets, de longueur 40 actuellement, contenant les chiffres 0,1,...,6), j'ai créé une "z-pile".

Faisons une digression maintenant afin d'introduire un mot évitant des confusions. Nous avons déjà deux piles. Trois peuvent devenir intolérables! Le mot auquel je fais allusion est un synonyme de "(". Il est défini simplement par

```
: (Z: ( -- <texte> )
    [ FIND ( ] LITERAL EXECUTE ; IMMEDIATE
```

Il sert à décrire le contenu de la z-pile.

Revenons à notre sujet. J'ai besoin d'entrer des z-numéraux. Celà se fait avec

```
Z ( -- znum ) (Z: -- z )
```

(prend un z-numéral dans le flot d'entrée, contrôle ses chiffres, et le place sur la z-pile s'il est valide).

Vous voyez ici l'utilité de mon délimiteur "(Z:". Addition et multiplication étant des opérations binaires, il est pratique de disposer du mot

```
: ZZ ( -- znum1 znum2 ) (Z: -- z1 z2 )
    Z Z ;
```

L'arithmétique des z-numéraux se fait avec

```
Z+ (Z: z1 z2 -- z3 ) ( z3 := z1 + z2 )
Z* (Z: z1 z2 -- z3 ) ( z3 := z1 * z2 )
Z- (Z: z1 z2 -- z3 ) ( z3 := z1 - z2 )
ZNEGATE (Z: z1 -- z2 ) ( z2 := - z1 )
ZD* (d -- ) (Z: z1 -- z2 ) ( z2 := z1 * d)
ZD/ (d -- ) (Z: z1 -- z2 ) ( z2 := z1 / d)
Z10* (Z: z1 -- z2 ) ( z2 := z1 * 10 )
Z10/ (Z: z1 -- z2 ) ( z2 := z1 / 10 )
```

où d représente un chiffre et β = 3 ou 2+Ω est la base.

Pour manipuler la z-pile, j'imite les mots FORTH standard. Ainsi

```
ZDUP (Z: z -- z z )
2ZDUP (Z: z1 z2 -- z1 z2 z1 z2 )
ZDROP (Z: z -- )
ZSWAP (Z: z1 z2 -- z2 z1 )
ZOVER (Z: z1 z2 -- z1 z2 z1 )
ZROT (Z: z1 z2 z3 -- z2 z3 z1 )
```

Pour les sorties :

```
Z. (Z: z -- )
ZG. (Z: z -- )
```

le premier mot affiche la représentation numérale de z et le dernier sa représentation gaussienne, d'après la définition

$$z = \sum_{0}^{n} d_i \beta^i = [d_n \ldots d_1 d_0]_\beta$$

Vous voyez maintenant que mon système FORTH pour l'arithmétique d'entiers équilibrés est essentiellement complète, à une exception notable près (qui n'est pas une exception en théorie). Je ne sais pas, actuellement, comment définir correctement les mots

```
Z/ (Z: z1 z2 -- z3 )
ZMOD (Z: z1 z2 -- z3 )
Z/MOD (Z: z1 z2 -- z3 z4 )
```

Avant de voir la division, veuillez regarder la Figure 2. Cette page montre un exercice typique sur mon système. Je démontre, par exemple, que

145 + 115 = 260 et 145 x 115 = 16675

ou bien, si vous êtes vraiment branché, que

122101 + 11121 = 101202 et

122101 x 11121 = 1022020221

Je montre que, quels que soient vos efforts, vous oubliez toujours dans quelle base vous vous trouvez si elle n'est pas décimale. Et quand vous ne vous souvenez plus de ce qui est empilé, vous pouvez aussi bien abandonner.

Figure 2.

```
TRIT OK
2Z 122101 11121 OK
ZSWAP 2ZDUP CR ZG.  4 SPACES ZG.
RE:  145 IM: 0  RE:  115 IM:  0 OK
2ZDUP Z+ ZSWAP ZROT Z* ZSWAP OK
ZDUP CR Z.  4 SPACES ZG.  ZDUP CR Z. 4 SPACES ZG.
101202      RE: 260 IM: 0
1022020221 RE:  16675 IM: 0 OK
Z  101202  OK
ZDUP Z10* Z10* Z10* OK
ZSWAP Z10/ Z10/ ZSWAP OK
ZDUP CR Z. 4 SPACES ZG. ZDUP CR Z. 4 SPACES ZG.
101202000   RE: 7020  IM:  0
101    RE:  10 IM:  0 OK
QUINT OK
2Z 31122 12345 5 IS NOT A DIGIT ?
2Z 31122 12340 OK
ZSWAP 2ZDUP CR ZG. 4 SPACES ZG.
RE:   11 IM:  -6  RE: -20 IM: 20 OK
2ZDUP Z+ ZSWAP ZROT Z* ZSWAP OK
ZDUP CR Z. 4 SPACES ZG. ZDUP CR Z. 4 SPACES ZG.
13002  RE: -9 IM: 14
41430210   RE:  -100 IM: 340 OK
Z 12340 OK
2 Z0* 2 Z0* 2 Z0* 2 Z0* ZDUP Z. 12340 OK
ZDUP 2 Z0* ZDUP 2 Z0* ZDUP 2 Z0* OK
CR ZG. 4 SPACES ZG. CR ZG. 4 SPACES ZG.
RE:  20 IM:  20   RE:  20 IM:  -20
RE:  -20 IM:  -20   RE:  -20 IM:  20 OK
SEPT OK
2Z 22345 66035 OK
ZSWAP 2ZDUP CR ZG. 4 SPACES ZG.
RE:  -83   IM:  59    RE:  55 IM:  16 OK
2ZDUP Z+ ZSWAP ZROT Z* ZSWAP OK
ZDUP CR Z. 4 SPACES ZG. ZDUP CR Z. 4 SPACES ZG.
12623   RE: -28  IM:  75
1506000243  RE:  -5509:  IM:  2861 OK
Z 66035 OK
ZDUP 2 Z0/ ZDUP 2 Z0/ ZDUP 2 Z0/ ZDUP 2 Z0/
     ZDUP 2 Z0/ OK
CR ZG. 4 SPACES ZG. 4 SPACES ZG. CR ZG.
    4 SPACES ZG. 4 SPACES ZG.
RE:  -16 IM: 71 RE: -71  IM: 55 RE: -55 IM: -16
RE: 16   IM: -71 RE: 71  IM: -55 RE: 55 IM: 16 OK
Z10* Z- STACK UNDERFLOW ?
```

Pour souligner ce que je vais faire maintenant, faisons un calcul. D'après la Fig. 2

$$22345 = -83 + 59\Omega,$$

$$66035 = 55 + 16\Omega$$

et leur produit est

$$1506000243 = -5509 + 2861\Omega,$$

dans le système septénaire équilibré. Souvenez-vous que la partie imaginaire n'a pas ici la forme usuelle! Le symbole Ω représente $(1 + \sqrt{-3})/2$ et non $\sqrt{-1}$.

Je vérifie maintenant le produit en utilisant les représentations gaussiennes et le fait que $\Omega^2 = \Omega - 1$:

$$\begin{array}{l} \quad -83 + 59\,\Omega \\ \underline{\times \quad 55 + 16\,\Omega} \\ -55 \times 83 + (55 \times 59 - 16 \times 83)\,\Omega + 16 \times 59\,\Omega^2 \\ = -4565 + 1917\,\Omega + 944\,(\Omega - 1) \\ = -(4565 + 944) + (1917 + 944)\,\Omega \\ = -5509 + 2861\,\Omega \end{array}$$

comme attendu (du moins par moi).

7 - Division.

Etant donnés deux numéraux a, b, avec b<>0,
trouvez des numéraux q, c tels que
(1) a = qb + c ,
(2) N(c) < N(b).

J'ai décrit la norme gaussienne N pour certains entiers algébriques en Section 4. Cependant, la "norme" N dans l'énoncé du problème de la division n'a pas à être (du moins, dans notre cas, ne devrait pas être) la norme gaussienne.

La notion rigoureuse de norme a été décrite par Th. Motzkin [3]. Une norme N doit prendre des numéraux non nuls pour des valeurs d'entiers non négatives. Elle doit satisfaire N(b) ≤ N(a) tant que b divise a . Si b ne divise pas a, alors il doit exister des numéraux q et c satisfaisant (1) et (2). Le véritable problème de la division est de trouver une norme N appropriée.

Voici un exemple. Si les numéraux sont des numéraux binaires pour des entiers (plus le bit de signe), alors N peut être la longueur numérale. Dans un sens décrit par Motzkin, c'est la norme la plus efficace pour des entiers rationnels. Mais 2 est très spécial ici. La longueur numérale dans toute base plus grande n'est plus une norme.

Le problème avec la norme en base 2 est qu'elle est inadaptée à mes buts. Le système ternaire équilibré a une base 3.

Considérons alors la fonction entière

$N(a) = 2n$ si $(3^n+1)/2 \le |a| < (5.3^{n-1}+1)/2$
$N(a) = 2n+1$ si $(5.3^{n-1}+1)/2 \le |a| < (3^{n+1}+1)/2$
pour $|a| > 1$ et $n = 1, 2, \ldots,$
avec $N(\pm 1) = 1$, et $N(0) = 0$.

Cette fonction satisfait à toutes les contraintes de Motzkin pour une norme et convient à la base 3. Sur les numéraux ternaires équilibrés

$a = [a_n \ldots a_1 a_0]_3$
avec $a_n <> 0$
$N(a) = 2n$ si $a_n + a_{n-1} = 0$
$N(a) = 2n + 1$ sinon

Voici un algorithme de division dans le système ternaire équilibré basé sur cette norme N. Les numéraux a et b étant donnés, avec b <> 0 :

pour calculer le quotient q, le reste r

```
q := 0 ; c := a ;
WHILE (N(b) ≤ N(c)) DO
BEGIN {invariant de boucle: a = q*b + c}
   t := - lead(c)/lead(b) ;
   n := (N(c) - N(b))/2 ;
   q := q - t * β ** n
   c := c + b * t * β ** n
END.
```

La fonction "lead" dans cet algorithme retourne le chiffre de plus haut degré du numéral. Il est synonyme de "signe" dans le système ternaire équilibré. La division dans la première ligne du bloc BEGIN est une division par chiffre; dans la seconde ligne, une division entière.

A titre d'exemple de division dans le système ternaire équilibré, je vais diviser 1210211011 par 11201. Les quotients accumulés et les restes apparaissent directement sous les lignes horizontales. Les normes du diviseur et des restes apparaissent entre parenthèses.

(NdT : la forme anglo-saxonne de la division a été conservée.)

DIVISEUR		RESTES		QUOTIENTS
11201	(9)	1210211011	(18)	0
		22102		10000
		101201011	(17)	10000
		22102		10000
		2221011	(13)	120000
		11201		200
		211111	(10)	120200
		11201		2
		2122	(6)	120202

Ce calcul montre que

1210211011 = 120202 x 11201 + 2122

En notation décimale plus usuelle :

15178 = 152 x 100 - 22

Assez curieusement, le même algorithme et la même norme , s'appliquent presque au système quinaire équilibré. Presque, mais pas tout à fait : voilà où j'en suis du problème de la division.

8 - Conclusion.

Bien que je ne prétende pas que mon exercice ait beaucoup d'importance pratique (mais il le pourrait), je croit vraiment qu'il représente une application intéressante de FORTH. Si vous êtes intéressé par l'histoire du système ternaire équilibré consultez Knuth (2, pp. 190-192). Gilbert (1) fait un bel exposé sur les systèmes de nombres complexes utilisant les entiers.

Références :

[1] Gilbert, W. J.: Arithmetic in complex bases. Mathematics Magazine 57 (2, mars 1984), 77-81.

[2] Knuth, D. E.: The art of computer programming, vol.2/Seminum. Algorithms. Addison-Wesley, 1981.

[3] Motzkin, Th.: The Euclidean Algorithm. Bull. Am. Math. Soc. 55 (12, dec. 1949).

1984, Rochester Forth Conference.
####### trad. A. JACCOMARD, nov. 1986. ######

NOTE DE LA REDACTION: Suite à la parution de cet article, nous nous attendons recevoir de nombreux appels et courriers pour obtenir des précisions concernants les références. Aussi nous permettons-nous de prendre les devants en vous avertissant que cet article est avant tout une traduction, lesdites références n'étant pas en notre possession. Cependant, s'il se trouve un adhérent les ayant en sa possession, nous lui serions gré de nous les communiquer.

D'autre part, cet article n'est accompagné d'aucun programme d'application pratique, et ce bien que l'auteur en fasse abondament référence. Aussi nous suggérons-vous, à titre d'exercice, de créer ce programme en respectant la syntaxe et les primitives telles qu'elles sont définies dans le présent article.

Bon courage..



Des règles complètes comprendraient en outre, les niveaux non traités ici, de nombreuses configurations de proximité entre deux ou trois niveaux, spécialement lorsque le précontact est l'un d'entre eux, les dimensions finies et leur tolérance etc...

EVOLUTION DU CONCEPT
DE REGLES DE DESSIN

Le dessin d'un circuit appelle une vérification du respect des règles précitées. Cette étape n'est plus envisageable manuellement et fait partie intégrante de l'arsenal CAO actuellement connu généralement sous le vocable DRC (Design Rules Checking), ces programmes peuvent être fiables mais toujours lourds à manipuler. Diverses approches tentent d'éviter ce contrôle à postériori tout en limitant le nombre de paramètres que doit manipuler le concepteur.

La première démarche est symbolique au niveau composants. Les composants élémentaires sont des transistors, des contacts, des connexions, etc... qui sont placés selon les règles de dessin généralement réduites. S'il est possible de quantifier ces règles en un petit nombre de multiples d'une dimension de base, on aboutit au "stick daigram" de Mead and Conway où la description peut être faite sur une grille figée. Si la gestion des règles de dessin est assurée par informatique, alors il est possible de dessiner sur une grille symbolique qui sera ensuite tassée par ordinateur, c'est le programme Tricky.

Une approche symbolique plus rationnelle consiste à aborder le niveau "porte". Les composants de base sont un nombre restreint de portes logiques et de conneions dont les dimensions sont quantifiées par rapport à une grille de base. Un exemple significatif est l'approche MDMOS où la largeur des portes est fixe, la longueur variable ainsi que la position des accès. Les risques d'incohérence logique ou électrique sont ainsi réduits au prix d'une légère perte de densité (environ 10%).

Au niveau supérieur, on peut citer l'approche cellulaire où les fonctions de base sont fixées topologiquement, électriquement et logiquement. Généralement une des dimensions est commune ou multiple d'un pas de base et la conception consiste à placer ces cellules et à les interconnecter entre elles. La sécurité électrique et logique devient alors très bonne au détriment de la densité (environ 30 %).

Le dernier stade qui est la frontière entre la conception de circuits et la conception de systèmes est le réseau prédiffusé. Un jeu de fonctions standard déjà placées est proposé au concepteur qui doit réaliser l'interconnexion grâce à un niveau de métallisation. Cette méthode se prête à une automatisation poussée et conduit à une sécurité absolue sur les plans logiques et électriques, au prix d'un choix restreint de fonctions de base.

Ces diverses approches ne sont pas à considérer comme concurrentes, car chacune a l'intérêt de proposer un compromis différent sur le plan du temps de conception, du coût du circuit et de ses performances. L'avenir appartient peut-être aux compilateurs de silicium, programmes qui permettraient d'obtenir les masques à partir d'une spécification formelle de la fonction à réaliser.